<< RETURN

Verlet Integration and Constraints in a Six Degree of Freedom Rigid Body Physics Simulation

Rick Baltman
Ron Radeztsky Jr.
Rainbow Studios, Inc.
4001 North 3rd Street, Suite 310
Phoenix, Arizona 85012

rickb@rainbowstudios.com
ronr@rainbowstudios.com
www.rainbowstudios.com

Abstract

At the 1999 Game Developers Conference, Thomas Jakobsen presented a semi-implicit rigid body physics simulation. The simulation could integrate stiff systems at the time steps used in games without the need to compute expensive Jacobian matrices and had very simple constraint implementations. However, the simulation was limited because it did not include a velocity state and it did not integrate the rotational equations of motion, necessitating large numbers of particles and distance constraints to simulate a single rigid body. It was also difficult to set up pin constraints, common in rag-doll simulations.

By extending the simulation to six degrees of freedom through the Verlet-like integration of a state quaternion, joint constraints became simple and straightforward, though a slightly more complicated constraint equation was required. Two methods for implementing the additional degrees of freedom are presented; one that uses the current quaternion and last quaternion as the system states and one that uses the quaternion and the angular rate. The advantages and disadvantages of both methods are discussed. The equations used in pin and angle constraints and the method for resolving collisions are presented. This simulation was developed for the game MX Unleashed, published by THQ.

1 Introduction

The decision to use the simulation method described in this paper was made by discovering the disadvantages of most of the simulation techniques that have already been published. Initially, a simulation using explicitly integrated equations of motion was developed. This simulation had the familiar stability problems when trying to integrate stiff systems. Constraints were formulated using the methods described by [Witkin]; write the constraint equation, differentiate it twice and solve for the constraint forces. The constraint solver worked, but some of the constraints had mathematically indeterminate solutions at specific orientations of the rigid bodies that had to be carefully avoided. Collision response was handled through the use of impulses [Baraff] since penalty methods could not be made stiff enough. Unfortunately, the rigid body objects would pop once they had come to rest on a surface.

Implicit integrators were used in order to handle stiffer systems. However, it was next to impossible to derive an expression for the Jacobian, or force derivative matrix, for some of the most computationally intensive, non-linear force models, like the tire model in a car simulation. Instead, the Jacobian was computed numerically by calling these force models a couple of times with small perturbations in each of the states. The cost of doing this, combined with the cost of solving a large set of linear equations, became prohibitive, especially when trying to simulate many four wheeled vehicles.

The simulation presented by [Jakobsen] was implemented next. The semi-implicit Verlet-like integrator permitted the use of springs stiff enough to handle resting contacts without the need to compute a Jacobian and the simple constraints always worked. However, since the simulation did not include a velocity state and did not integrate the rotational degrees of freedom, it had to be extended to be practical in our application. The final result met all of our needs and is described herein.

2 The Particle

The basic particle is represented as a point of mass, m, that has a position, $\mathbf{x}(t)$ and a velocity, $\mathbf{v}(t)$, that are functions of time.

3 The Rigid Body

For the purposes of this simulation, a rigid body is approximated by a particle with a mass distribution, or inertia, $\mathbf{I}$, that has orientation, represented by a quaternion, $\mathbf{q}(t)$.

$$\mathbf{I} = \begin{bmatrix} I_{xx} & I_{xy} & I_{xz} \\ I_{yx} & I_{yy} & I_{yz} \\ I_{zx} & I_{zy} & I_{zz} \end{bmatrix} \tag{3.1}$$

$\mathbf{I}$ is not a function of time. It is computed once in the rigid body coordinate frame. This saves the cost of rotating the inertia matrix into the world coordinate frame each time step. The complete inertia matrix, $\mathbf{I}$, can be used in the equations of motion that follow. However, by setting all of the products of inertia, or off-diagonal terms, to zero, the matrix can be replaced with a vector, Equation (3.2), which reduces the number of calculations that must be performed throughout the simulation.

$$\mathbf{I} = \begin{bmatrix} I_{xx} \\ I_{yy} \\ I_{zz} \end{bmatrix} \tag{3.2}$$

4 Equations of Motion

To animate the particle, the simulation presented by [Jakobsen] was used. However, since the simulation only integrated the positions of particles, sets of particles had to be connected together with distance constraints in order to simulate a rigid body. Consequently, it was more expensive than including the rotational degrees of freedom.

The [Jakobsen] integrator was modified slightly to introduce a damping term:

$$\begin{aligned} \mathbf{x}(t+\Delta t) &= \left(1+\xi_{velocity}\right)\mathbf{x}(t) - \xi_{velocity}\,\mathbf{x}(t-\Delta t) + \frac{1}{m}\mathbf{F}\Delta t^2 \\ \mathbf{x}(t-\Delta t) &= \mathbf{x}(t) \end{aligned} \tag{4.1}$$

The $\xi_{velocity}$ term can be used to introduce artificial damping into these equations of motion. Normally, $\xi_{velocity}$ equals one, and this equation is identical to [Jakobsen]. When the damping constant is zero, the equations of motion are reduced to $\mathbf{F} = m\mathbf{v}$; as soon as the external force is removed, the body stops moving.

This method was extended to integrate the rotational degrees of freedom:

$$\begin{aligned} \boldsymbol{\delta}\mathbf{q} &= \mathrm{Slerp}\left(\hat{\mathbf{q}}, \mathbf{q}(t)\overline{\mathbf{q}}(t-\Delta t), \xi_{angular\,rate}\right) \\ \mathbf{q}'(t+\Delta t) &= \boldsymbol{\delta}\mathbf{q}\,\mathbf{q}(t) \\ \mathbf{q}(t+\Delta t) &= \mathbf{q}(t) + \dot{\mathbf{q}}(\mathbf{q}'(t+\Delta t), \mathbf{I}^{-1}\boldsymbol{\tau}(t))\Delta t^2 \\ \mathbf{q}(t-\Delta t) &= \mathbf{q}(t) \end{aligned} \tag{4.2}$$

where

$$\dot{\mathbf{q}}(\mathbf{q},\boldsymbol{\omega}) = \frac{1}{2}\mathbf{q}\begin{bmatrix} 0 \\ \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} \tag{4.3}$$

Like $\xi_{velocity}$ in the translational equations of motion, $\xi_{angular\ rate}$ normally equals one. If the angular rate damping equals zero, the body will stop rotating when the external torque, $\boldsymbol{\tau}(t)$, is removed. If $\xi_{angular\ rate}$ is always one, the Slerp can be skipped and $\boldsymbol{\delta}\mathbf{q} = \mathbf{q}(t)\,\overline{\mathbf{q}}(t\text{-}\Delta t)$. The Slerp, or spherical linear interpolation function, along with some of the other quaternion math operations are listed in Appendix A. Equation (4.3) is reproduced from [Refson].

Initially, this worked well. However, by using the current state and the last state in the equations of motion, any force that is a function of the velocity or angular rate, like a damping term, must use a finite difference method to compute them.

$$\mathbf{v} = \frac{\left(\mathbf{x}(t) - \mathbf{x}(t-\Delta t)\right)}{\Delta t} \tag{4.4}$$

$$\begin{aligned} \boldsymbol{\omega}_{world} &= \frac{\text{ComputeAngleAxis}\left(\mathbf{q}(t)\overline{\mathbf{q}}(t-\Delta t)\right)}{\Delta t} \\ \boldsymbol{\omega} &= \overline{\mathbf{q}}(t)\boldsymbol{\omega}_{world}\mathbf{q}(t) \end{aligned} \tag{4.5}$$

These calculations can be very noisy and the problem is exacerbated when constraints come into play and instantaneously change the particle states. (Please note that $\boldsymbol{\omega} = \overline{\mathbf{q}}(t)\ \boldsymbol{\omega}_{world}\ \mathbf{q}(t)$ simply rotates the angular rate from world to body coordinates.)

This problem is resolved in the translational degrees of freedom by replacing the $\mathbf{x}(t\text{-}\Delta t)$ state with velocity, $\mathbf{v}(t)$. The new equations of motion become:

$$\begin{aligned} \mathbf{x}(t+\Delta t) &= \mathbf{x}(t) + \mathbf{v}(t)\Delta t + \frac{1}{2}\frac{\mathbf{F}}{m}\Delta t^2 \\ \mathbf{v}(t+\Delta t) &= \xi_{velocity}\mathbf{v}(t) + \frac{\mathbf{F}}{m}\Delta t \end{aligned} \tag{4.6}$$

[Garberoglio] provides a description of a Verlet-like integrator for quaternions. Like the translational equations of motion, the $\mathbf{q}(t\text{-}\Delta t)$ state is replaced with body angular rate, $\boldsymbol{\omega}(t)$.

$$\begin{aligned} \mathbf{q}(t+\Delta t) &= \mathbf{q}(t) + \dot{\mathbf{q}}\left(\mathbf{q}(t), \boldsymbol{\omega}(t)\right)\Delta t + \frac{1}{2}\ddot{\mathbf{q}}\left(\mathbf{q}(t), \boldsymbol{\omega}(t), \mathbf{I}^{-1}\boldsymbol{\tau}(t)\right)\Delta t^2 \\ \boldsymbol{\omega}(t+\Delta t) &= \xi_{angular\,rate}\boldsymbol{\omega}(t) + \mathbf{I}^{-1}\boldsymbol{\tau}(t)\Delta t \end{aligned} \tag{4.7}$$

where

$$\ddot{\mathbf{q}}(\mathbf{q}, \boldsymbol{\omega}, \dot{\boldsymbol{\omega}}) = \frac{1}{2}\mathbf{q}\begin{bmatrix} -2\dot{\mathbf{q}}\cdot\dot{\mathbf{q}} \\ \dot{\omega}_x \\ \dot{\omega}_y \\ \dot{\omega}_z \end{bmatrix} \tag{4.8}$$

Although this second set of equations provides smooth velocities and angular rates, the constraints must now deal with these new states. In the first set, referred to as the velocity-less equations of motion, although constraints can set the velocity and angular rate by setting $\mathbf{x}(t\text{-}\Delta t)$ and $\mathbf{q}(t\text{-}\Delta t)$, they do not have to. At a minimum, they can simply reposition or reorient the body. Therefore, the set of equations used depends on the application. If the application does not generate forces that are a function of the rates, the velocity-less equations of motion can be used to take advantage of the very simple and fast constraint equations. Equation (4.8) is derived from [Refson].

5 Collision Response

The simulation uses the [Baraff] method for resolving collisions between two rigid bodies. However, instead of computing the contact forces for resting contacts, which can be expensive when there are multiple contact points, an easier and faster penalty method is used. This penalty method can be problematic when using explicit integration, but the semi-implicit integrator permits the use of stiff springs.

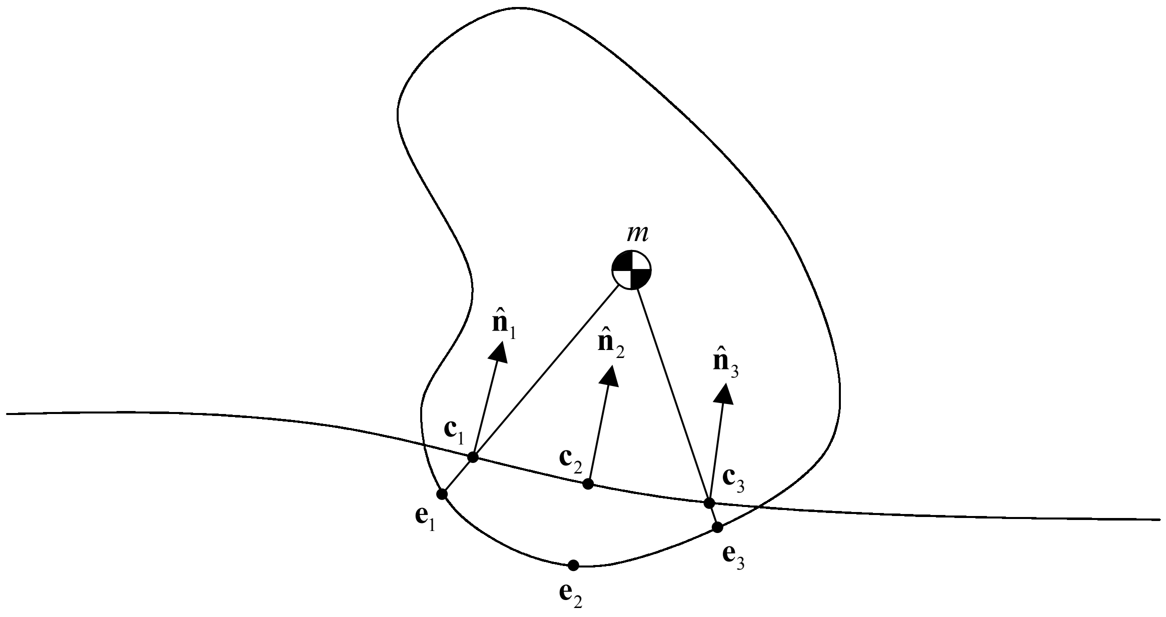

Figure 1 - Collision Geometry Output from Collision Detection Algorithm

As illustrated in Figure 1, the collision detection algorithm returns a set of contact points and surface normals. For every contact point, $\mathbf{c}_i$, and surface normal, $\hat{\mathbf{n}}_i$, the following set of calculations is performed. First the velocity at the point of contact is computed.

$$\begin{aligned} \mathbf{r}_i^a &= \mathbf{c}_i - \mathbf{x}^a(t) \\ \boldsymbol{\omega}_{world}^a &= \mathbf{q}^a(t)\boldsymbol{\omega}^a(t)\overline{\mathbf{q}}^a(t) \\ \mathbf{v}_{contact}^a &= \mathbf{v}^a(t) + \boldsymbol{\omega}_{world}^a \times \mathbf{r}_i^a \end{aligned} \tag{5.1}$$

If a second rigid body is involved in the collision, the second body's contact point velocity is computed using Equation (5.2), otherwise it is set to zero.

$$\begin{aligned} \mathbf{r}_i^b &= \mathbf{c}_i - \mathbf{x}^b(t) \\ \boldsymbol{\omega}_{world}^b &= \mathbf{q}^b(t)\,\boldsymbol{\omega}^b(t)\,\overline{\mathbf{q}}^b(t) \\ \mathbf{v}_{contact}^b &= \mathbf{v}^b(t) + \boldsymbol{\omega}_{world}^b \times \mathbf{r}_i^b \end{aligned} \tag{5.2}$$

The relative velocity is simply:

$$\begin{aligned} \mathbf{v}_{relative}^a &= \mathbf{v}_{contact}^a - \mathbf{v}_{contact}^b \\ \mathbf{v}_{relative}^b &= -\mathbf{v}_{relative}^a \end{aligned} \tag{5.3}$$

If the relative velocity along the surface normal is small, the contact point is considered a resting contact.

$$\begin{aligned} s_{relative}^a &= \mathbf{v}_{relative}^a \cdot \hat{\mathbf{n}}_i^a \\ s_{relative}^b &= -s_{relative}^a \end{aligned} \tag{5.4}$$

In which case, a penalty force is calculated so the two objects can rest on one another.

$$\begin{aligned} d_{penetration}^a &= \left(\mathbf{c}_i - \mathbf{e}_i^a\right) \cdot \hat{\mathbf{n}}_i^a \\ \dot{d}_{penetration}^a &= \mathbf{v}_{relative}^a \cdot \hat{\mathbf{n}}_i^a \\ \mathbf{F}_{penetration}^a &= \left(k_{spring}^a d_{penetration}^a - k_{damping}^a \dot{d}_{penetration}^a\right) \hat{\mathbf{n}}_i^a \end{aligned} \tag{5.5}$$

$\mathbf{e}_i$ is the point on the surface of the rigid body that also lies on the ray between the center of gravity and the contact point, $\mathbf{c}_i$ (Figure 1). If a second rigid body is involved in the collision, this force is *not* applied to the second body. Instead, a second penetration force is calculated using a different set of spring and damping constants and both forces are scaled in relation to their masses, as presented in Equation (5.7). These forces are then applied to their respective rigid bodies.

$$\begin{aligned} d_{penetration}^b &= \left(\mathbf{c}_i - \mathbf{e}_i^b\right) \cdot \hat{\mathbf{n}}_i^b \\ \dot{d}_{penetration}^b &= \mathbf{v}_{relative}^b \cdot \hat{\mathbf{n}}_i^b \\ \mathbf{F}_{penetration}^b &= \left(k_{spring}^b d_{penetration}^b - k_{damping}^b \dot{d}_{penetration}^b\right) \hat{\mathbf{n}}_i^b \end{aligned} \tag{5.6}$$

$$\begin{aligned} \mathbf{F}_{penetration}^a &= \min\left(\frac{m^b}{m^a}, 1\right) \mathbf{F}_{penetration}^a \\ \mathbf{F}_{penetration}^b &= \min\left(\frac{m^a}{m^b}, 1\right) \mathbf{F}_{penetration}^b \end{aligned} \tag{5.7}$$

A potential problem with this approach to collision resolution is that the resulting motion of the rigid body or bodies varies with the number of contact points returned from the collision detection algorithm. More points generate more resting and friction forces so k_{spring} must be lowered to compensate. The collision detection algorithm used with this simulation always returns the same number of contact points for any particular collision orientation so this was not a problem.

If the penetration speed, $s_{relative}$, is large, impulses are used to instantaneously change the velocity and angular rate of the objects involved in the collision. Other than minor changes in notation, Equation (5.9) is presented unchanged from [Barraf] page D47, Equation (8-18).

$$\begin{aligned}
\mathbf{r}_{i\,body}^{a} &= \overline{\mathbf{q}}^{a}(t)\,\mathbf{r}_{i}^{a}\,\mathbf{q}^{a}(t) \\
\hat{\mathbf{n}}_{i\,body}^{a} &= \overline{\mathbf{q}}^{a}(t)\,\hat{\mathbf{n}}_{i}^{a}\,\mathbf{q}^{a}(t) \\
\mathbf{r}_{i\,body}^{b} &= \overline{\mathbf{q}}^{b}(t)\,\mathbf{r}_{i}^{b}\,\mathbf{q}^{b}(t) \\
\hat{\mathbf{n}}_{i\,body}^{b} &= \overline{\mathbf{q}}^{b}(t)\,\hat{\mathbf{n}}_{i}^{b}\,\mathbf{q}^{b}(t)
\end{aligned} \tag{5.8}$$

$$j = \frac{-(1+\varepsilon)\,s_{relative}^{a}}{\frac{1}{m^{a}} + \frac{1}{m^{b}} + \hat{\mathbf{n}}_{i\,body}^{a}\cdot\left(\mathbf{I}^{a^{-1}}\left(\mathbf{r}_{i\,body}^{a}\times\hat{\mathbf{n}}_{i\,body}^{a}\right)\right)\times\mathbf{r}_{i\,body}^{a} + \hat{\mathbf{n}}_{i\,body}^{b}\cdot\left(\mathbf{I}^{b^{-1}}\left(\mathbf{r}_{i\,body}^{b}\times\hat{\mathbf{n}}_{i\,body}^{b}\right)\right)\times\mathbf{r}_{i\,body}^{b}} \tag{5.9}$$

ε is the coefficient of restitution. This impulse is then used to change the velocities and angular rates of the two colliding bodies.

$$\begin{aligned}
\mathbf{v}_{new}^{a}(t) &= \mathbf{v}^{a}(t) + \frac{j}{m^{a}}\hat{\mathbf{n}}_{i}^{a} \\
\boldsymbol{\omega}_{new}^{a}(t) &= \boldsymbol{\omega}^{a}(t) + \mathbf{I}^{a^{-1}}\left(\mathbf{r}_{i\,body}^{a}\times j\hat{\mathbf{n}}_{i\,body}^{a}\right) \\
\mathbf{v}_{new}^{b}(t) &= \mathbf{v}^{b}(t) + \frac{j}{m^{b}}\hat{\mathbf{n}}_{i}^{b} \\
\boldsymbol{\omega}_{new}^{b}(t) &= \boldsymbol{\omega}^{b}(t) + \mathbf{I}^{b^{-1}}\left(\mathbf{r}_{i\,body}^{b}\times j\hat{\mathbf{n}}_{i\,body}^{b}\right)
\end{aligned} \tag{5.10}$$

One may question the need for Equation (5.8) if the inertia matrix, $\mathbf{I}^{-1}$, were transformed into the world coordinate frame for each of the rigid bodies. This savings comes at the cost of transforming the inertia matrix, $\mathbf{I}$, into the world coordinate frame and then finding its inverse, which is more expensive than four vector rotations. Some simulations transform the inertia matrix into the world frame after the equations of motion are integrated. The problem with this approach is that $\mathbf{I}$ and its inverse, $\mathbf{I}^{-1}$, would have to be re-transformed every time the orientation of the rigid body is updated by a constraint.

The final step of the collision response is to compute a friction force using a simple kinetic friction model. This model transitions into a viscous friction model at low sliding speeds and as a result, ignores the effects of static friction. This was suitable for our purposes, but any friction model can be used. The friction coefficient, $\mu_{kinetic}$, changes based on which surface the rigid body is sliding on.

$$\begin{aligned}
\mathbf{v}_{perpendicular} &= \left(\mathbf{v}_{relative} \cdot \hat{\mathbf{n}}_i\right)\hat{\mathbf{n}}_i - \mathbf{v}_{relative} \\
\mathbf{F}_{friction} &= \mu_{kinetic}\left|\mathbf{F}_{penetration}\right|\hat{\mathbf{v}}_{perpendicular} \qquad (5.11) \\
\mathbf{F}_{friction} &= \min\left(\left|\mathbf{v}_{perpendicular}\right|,1\right)\mathbf{F}_{friction}
\end{aligned}$$

6 The Distance Constraint

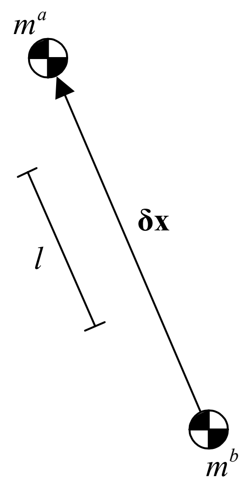

Figure 2 - Particle Distance Constraint

The distance between two particles can be kept relatively constant through the use of the distance constraint (Figure 2). This is very useful in rope, cloth, water surface and soft body simulations. The simple distance constraint presented by [Jakobsen], page 388, assumes that the masses of the two particles are equal, which is generally true. However, to include the effects of mass, it is simply a matter of repositioning the particles as a function of their mass ratios.

$$c = \left|\mathbf{x}^a(t) - \mathbf{x}^b(t)\right| - l = 0 \qquad (6.1)$$

$$\begin{aligned}
&\boldsymbol{\delta}\mathbf{x} = \mathbf{x}^a(t) - \mathbf{x}^b(t) \\
&\mathbf{x}_{error} = k_{strength}\left(l - |\boldsymbol{\delta}\mathbf{x}|\right)\boldsymbol{\delta}\hat{\mathbf{x}} \\
&\mathbf{x}^a_{new}(t) = \mathbf{x}^a(t) + \frac{\frac{1}{m^a}}{\frac{1}{m^a} + \frac{1}{m^b}}\mathbf{x}_{error} \\
&\mathbf{x}^b_{new}(t) = \mathbf{x}^b(t) - \frac{\frac{1}{m^b}}{\frac{1}{m^a} + \frac{1}{m^b}}\mathbf{x}_{error}
\end{aligned} \tag{6.2}$$

If the masses are identical, the mass ratio terms can simply be replaced with ½. A constraint strength term, $k_{strength}$, which can range from zero to one, is introduced to soften the distance constraint. This allows the rope, cloth or soft body to stretch more than is possible by only lowering the number of constraint iterations.

In games, the results are more than satisfactory. However, by ignoring the velocity states, residual motion is left in the particles that may not be desirable. Differentiating the constraint equation, Equation (6.1), yields Equation (6.3); the velocity difference between the two particles cannot move the particles closer or farther apart.

$$\dot{c} = \left(\mathbf{v}^a(t) - \mathbf{v}^b(t)\right) \cdot \boldsymbol{\delta}\hat{\mathbf{x}} = 0 \tag{6.3}$$

$$\begin{aligned}
&\boldsymbol{\delta}\mathbf{v} = \mathbf{v}^a(t) - \mathbf{v}^b(t) \\
&\mathbf{v}_{error} = k_{strength}\left(\boldsymbol{\delta}\mathbf{v} \cdot \boldsymbol{\delta}\hat{\mathbf{x}}\right) \\
&\mathbf{v}^a_{new}(t) = \mathbf{v}^a(t) + \frac{\frac{1}{m^a}}{\frac{1}{m^a} + \frac{1}{m^b}}\mathbf{v}_{error} \\
&\mathbf{v}^b_{new}(t) = \mathbf{v}^b(t) - \frac{\frac{1}{m^b}}{\frac{1}{m^a} + \frac{1}{m^b}}\mathbf{v}_{error}
\end{aligned} \tag{6.4}$$

If needed, these corrections can be made in the velocity-less equations of motion by replacing the expression for $\boldsymbol{\delta}\mathbf{v}$ in Equation (6.4) with the one in Equation (6.5) and then setting the previous positions of the two particles using Equation (6.6). However, if the velocity-less equations of motion are suitable for the application, these corrections are most likely unnecessary.

$$\boldsymbol{\delta}\mathbf{v} = \left(\mathbf{x}^b(t) - \mathbf{x}^b(t - \Delta t)\right) - \left(\mathbf{x}^a(t) - \mathbf{x}^a(t - \Delta t)\right) \tag{6.5}$$

$$\begin{aligned}\mathbf{x}^a(t-\Delta t) &= \mathbf{x}^a(t) - \mathbf{v}^a_{new}(t)\\ \mathbf{x}^b(t-\Delta t) &= \mathbf{x}^b(t) - \mathbf{v}^b_{new}(t)\end{aligned} \tag{6.6}$$

7 The Pin Constraint

The distance constraint becomes a pin constraint when it is applied to rigid bodies. It connects two bodies at arbitrary points in or on the surface as illustrated in Figure 3.

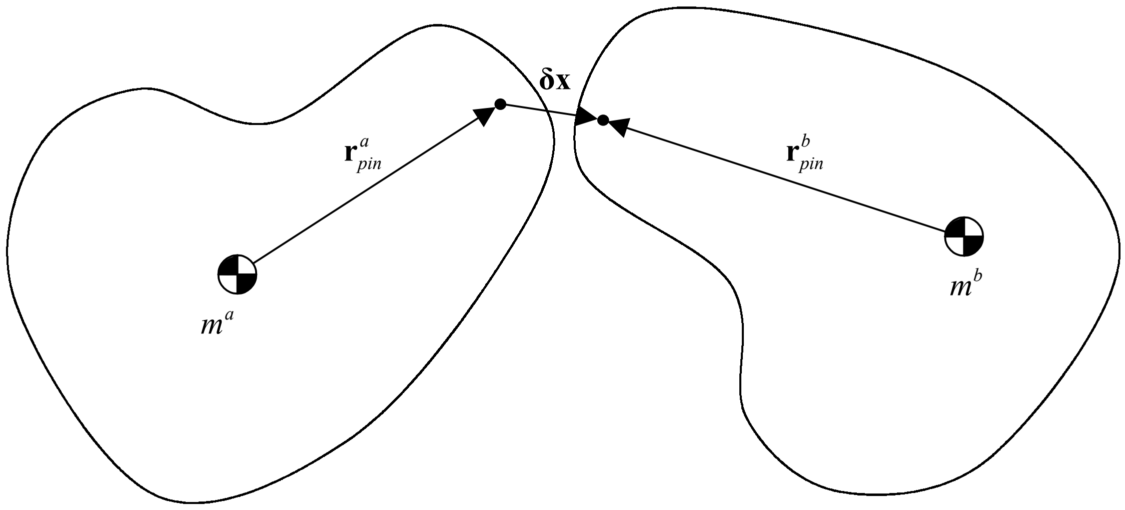

Figure 3 - Rigid Body Pin Constraint

The vector between the two pin points on the two bodies is first calculated:

$$\begin{aligned}\mathbf{x}^a_{pin} &= \mathbf{x}^a(t) + \mathbf{q}^a(t)\mathbf{r}^a_{pin}\overline{\mathbf{q}}^a(t)\\ \mathbf{x}^b_{pin} &= \mathbf{x}^b(t) + \mathbf{q}^b(t)\mathbf{r}^b_{pin}\overline{\mathbf{q}}^b(t)\\ \boldsymbol{\delta}\mathbf{x} &= \mathbf{x}^b_{pin} - \mathbf{x}^a_{pin}\end{aligned} \tag{7.1}$$

The problem is how to divide $\boldsymbol{\delta}\mathbf{x}$ between the orientations and positions of the two bodies and in what proportions. Fortunately, the impulse equation from [Baraff], Equation (5.9), was re-applied to arrive at the solution:

$$\begin{aligned}\boldsymbol{\delta}\hat{\mathbf{x}}^a_{body} &= \overline{\mathbf{q}}^a(t)\,\boldsymbol{\delta}\hat{\mathbf{x}}\,\mathbf{q}^a(t)\\ \boldsymbol{\delta}\hat{\mathbf{x}}^b_{body} &= \overline{\mathbf{q}}^b(t)\,\boldsymbol{\delta}\hat{\mathbf{x}}\,\mathbf{q}^b(t)\end{aligned} \tag{7.2}$$

$$j = \frac{-1}{\frac{1}{m^a} + \frac{1}{m^b} + \boldsymbol{\delta}\hat{\mathbf{x}}^a_{body}\cdot\left(\mathbf{I}^{a^{-1}}\left(\mathbf{r}^a_{pin}\times\boldsymbol{\delta}\hat{\mathbf{x}}^a_{body}\right)\right)\times\mathbf{r}^a_{pin} + \boldsymbol{\delta}\hat{\mathbf{x}}^b_{body}\cdot\left(\mathbf{I}^{b^{-1}}\left(\mathbf{r}^b_{pin}\times\boldsymbol{\delta}\hat{\mathbf{x}}^b_{body}\right)\right)\times\mathbf{r}^b_{pin}} \tag{7.3}$$

$$\begin{aligned}
\mathbf{x}^a_{new}(t) &= \mathbf{x}^a(t) + \frac{j}{m^a}\boldsymbol{\delta}\mathbf{x} \\
\mathbf{j}^a_{body} &= \overline{\mathbf{q}}^a(t)\left(j\boldsymbol{\delta}\mathbf{x}\right)\mathbf{q}^a(t) \\
\boldsymbol{\delta}\mathbf{q}^a &= \text{ComputeQuaternion}\left(\mathbf{q}^a(t)\left(\mathbf{I}^{a^{-1}}\left(\mathbf{r}^a_{pin}\times\mathbf{j}^a_{body}\right)\right)\overline{\mathbf{q}}^a(t)\right) \\
\mathbf{x}^b_{new}(t) &= \mathbf{x}^b(t) - \frac{j}{m^b}\boldsymbol{\delta}\mathbf{x} \\
\mathbf{j}^b_{body} &= \overline{\mathbf{q}}^b(t)\left(j\boldsymbol{\delta}\mathbf{x}\right)\mathbf{q}^b(t) \\
\boldsymbol{\delta}\mathbf{q}^b &= \text{ComputeQuaternion}\left(\mathbf{q}^b(t)\left(\mathbf{I}^{b^{-1}}\left(\mathbf{r}^b_{pin}\times\mathbf{j}^b_{body}\right)\right)\overline{\mathbf{q}}^b(t)\right)
\end{aligned} \tag{7.4}$$

If the velocity states are being used, the quaternions are not updated at this point because they are needed to constrain the velocities, as follows. When using the velocity-less equations of motion, update the quaternions now using Equation (7.9) and the constraint is fully implemented.

The velocity and angular rate are constrained in much the same way.

$$\begin{aligned}
\mathbf{v}^a_{pin} &= \mathbf{v}^a(t) + \mathbf{q}^a(t)\left(\boldsymbol{\omega}^a(t)\times\mathbf{r}^a_{pin}\right)\overline{\mathbf{q}}^a(t) \\
\mathbf{v}^b_{pin} &= \mathbf{v}^b(t) + \mathbf{q}^b(t)\left(\boldsymbol{\omega}^b(t)\times\mathbf{r}^b_{pin}\right)\overline{\mathbf{q}}^b(t) \\
\boldsymbol{\delta}\mathbf{v} &= \mathbf{v}^a_{pin} - \mathbf{v}^b_{pin}
\end{aligned} \tag{7.5}$$

The impulse equation is used once again to distribute the velocity error between the two body's velocities and angular rates.

$$\begin{aligned}
\boldsymbol{\delta}\hat{\mathbf{v}}^a_{body} &= \overline{\mathbf{q}}^a(t)\,\boldsymbol{\delta}\hat{\mathbf{v}}\,\mathbf{q}^a(t) \\
\boldsymbol{\delta}\hat{\mathbf{v}}^b_{body} &= \overline{\mathbf{q}}^b(t)\,\boldsymbol{\delta}\hat{\mathbf{v}}\,\mathbf{q}^b(t)
\end{aligned} \tag{7.6}$$

$$j = \frac{-1}{\frac{1}{m^a} + \frac{1}{m^b} + \boldsymbol{\delta}\hat{\mathbf{v}}^a_{body}\cdot\left(\mathbf{I}^{a^{-1}}\left(\mathbf{r}^a_{pin}\times\boldsymbol{\delta}\hat{\mathbf{v}}^a_{body}\right)\right)\times\mathbf{r}^a_{pin} + \boldsymbol{\delta}\hat{\mathbf{v}}^b_{body}\cdot\left(\mathbf{I}^{b^{-1}}\left(\mathbf{r}^b_{pin}\times\boldsymbol{\delta}\hat{\mathbf{v}}^b_{body}\right)\right)\times\mathbf{r}^b_{pin}} \tag{7.7}$$

$$\begin{aligned}
\mathbf{v}^a_{new}(t) &= \mathbf{v}^a(t) + \frac{j}{m^a}\boldsymbol{\delta}\mathbf{v} \\
\boldsymbol{\omega}^a(t) &= \boldsymbol{\omega}^a(t) + \mathbf{I}^{a^{-1}}\left(\mathbf{r}^a_{pin}\times\left(\overline{\mathbf{q}}^a(t)\left(j\boldsymbol{\delta}\mathbf{v}\right)\mathbf{q}^a(t)\right)\right) \\
\mathbf{v}^b_{new}(t) &= \mathbf{v}^b(t) - \frac{j}{m^b}\boldsymbol{\delta}\mathbf{v} \\
\boldsymbol{\omega}^b(t) &= \boldsymbol{\omega}^b(t) - \mathbf{I}^{b^{-1}}\left(\mathbf{r}^b_{pin}\times\left(\overline{\mathbf{q}}^b(t)\left(j\boldsymbol{\delta}\mathbf{v}\right)\mathbf{q}^b(t)\right)\right)
\end{aligned} \tag{7.8}$$

The last step is to update the quaternions using the corrections computed in Equation (7.4).

$$\begin{aligned}\mathbf{q}^a(t) &= \boldsymbol{\delta}\mathbf{q}^a\mathbf{q}^a(t)\\ \mathbf{q}^b(t) &= \boldsymbol{\delta}\mathbf{q}^b\mathbf{q}^b(t)\end{aligned} \tag{7.9}$$

As one may notice, the implementation of the pin constraint for the velocity-based equations of motion is about twice as expensive as the pin constraint in the velocity-less equations of motion. The pin constraint is also much more expensive than the very simple distance constraint. It is therefore important to try and use only particles and distance constraints whenever possible. Still, a single pin constraint connecting two rigid bodies involves fewer operations than assembling an entire network of eight particles and twelve distance constraints that do the same thing.

8 The Angle Constraint

Another very important constraint to implement is the angle constraint. It limits the amount one rigid body can rotate relative to another as shown in Figure 4.

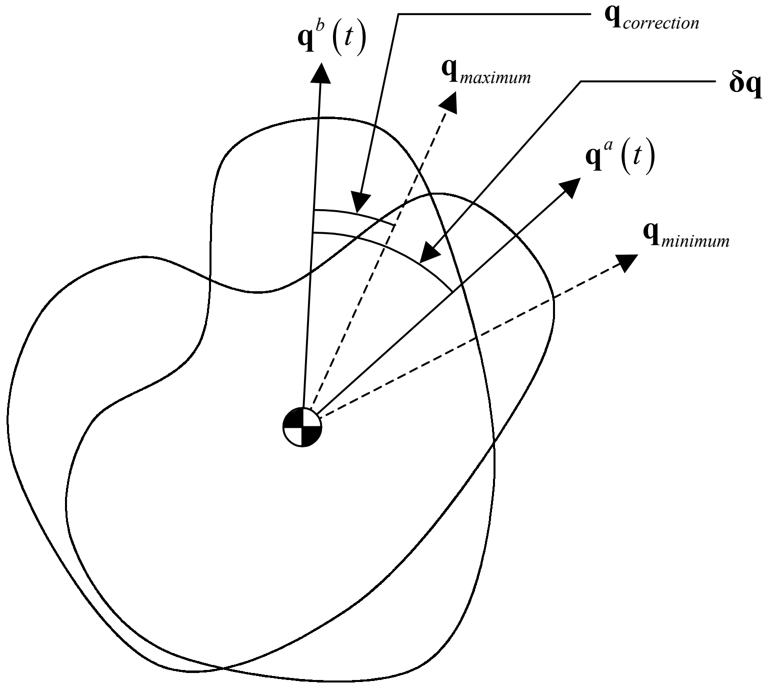

Figure 4 - Angle Constraint

Since the equations of motion are written in terms of quaternions, and orientation limits are intuitively expressed as angular rotations around an axis, $\boldsymbol{\theta}$, a simple conversion is made.

$$\mathbf{q}_{minimum} = \begin{bmatrix} 0 \\ \sin\frac{\theta_{minimum\,x}}{2} \\ \sin\frac{\theta_{minimum\,y}}{2} \\ \sin\frac{\theta_{minimum\,z}}{2} \end{bmatrix} \tag{8.1}$$

$$\mathbf{q}_{maximum} = \begin{bmatrix} 0 \\ \sin\frac{\theta_{maximum\,x}}{2} \\ \sin\frac{\theta_{maximum\,y}}{2} \\ \sin\frac{\theta_{maximum\,z}}{2} \end{bmatrix} \tag{8.2}$$

The orientation of the second body relative to the first is computed in the first body's coordinate frame:

$$\begin{aligned} \boldsymbol{\delta}\mathbf{q} &= \mathbf{q}^{b}(t)\,\overline{\mathbf{q}}^{a}(t) \\ \boldsymbol{\delta}\mathbf{q}^{a}_{body} &= \overline{\mathbf{q}}^{a}(t)\,\boldsymbol{\delta}\mathbf{q}\,\mathbf{q}^{a}(t) \end{aligned} \tag{8.3}$$

The rotation into the first rigid body coordinate frame is necessary because quaternions are always expressed in the world coordinate frame. However, the angle limits are expressed as rotations relative to the first rigid body. It is now a simple matter to limit the rotations:

$$\boldsymbol{\delta}\mathbf{q}^{a}_{body} = \begin{bmatrix} 0 \\ \min\left(\max\left(\delta q^{a}_{body\,x}, q_{minimum\,x}\right), q_{maximum\,x}\right) \\ \min\left(\max\left(\delta q^{a}_{body\,y}, q_{minimum\,y}\right), q_{maximum\,y}\right) \\ \min\left(\max\left(\delta q^{a}_{body\,z}, q_{minimum\,z}\right), q_{maximum\,z}\right) \end{bmatrix} \tag{8.4}$$

If none of the rotation limits have been exceeded, no further action is necessary. If an angle limit is violated, an orientation correction term is calculated in the world coordinate frame.

$$\delta q^{a}_{body\,w} = \sqrt{1 - \left|\boldsymbol{\delta}\mathbf{q}^{a}_{body}\right|} \tag{8.5}$$

$$\begin{aligned} \boldsymbol{\delta}\mathbf{q}' &= \mathbf{q}^{a}(t)\,\boldsymbol{\delta}\mathbf{q}^{a}_{body}\,\overline{\mathbf{q}}^{a}(t) \\ \mathbf{q}_{correction} &= \boldsymbol{\delta}\mathbf{q}\,\boldsymbol{\delta}\mathbf{q}' \end{aligned} \tag{8.6}$$

This correction quaternion, $\mathbf{q}_{correction}$, must now be divided between the two rigid bodies in proportion to their inertias about the axis of the correction.

$$\begin{aligned}
\boldsymbol{\theta}_{correction} &= ComputeAngleAxis\left(\mathbf{q}_{correction}\right) \\
\hat{\boldsymbol{\theta}}^{a}_{correction\ body} &= \overline{\mathbf{q}}^{a}\left(t\right)\hat{\boldsymbol{\theta}}_{correction}\mathbf{q}^{a}\left(t\right) \\
\hat{\boldsymbol{\theta}}^{b}_{correction\ body} &= \overline{\mathbf{q}}^{b}\left(t\right)\hat{\boldsymbol{\theta}}_{correction}\mathbf{q}^{b}\left(t\right) \\
I_{\theta^{a}} &= \hat{\boldsymbol{\theta}}^{a^{T}}_{correction\ body}\mathbf{I}^{a}\hat{\boldsymbol{\theta}}^{a}_{correction\ body} \\
I_{\theta^{b}} &= \hat{\boldsymbol{\theta}}^{b^{T}}_{correction\ body}\mathbf{I}^{b}\hat{\boldsymbol{\theta}}^{b}_{correction\ body}
\end{aligned} \tag{8.7}$$

$$\begin{aligned}
i^{a} &= \frac{\frac{1}{I_{\theta^{a}}}}{\frac{1}{I_{\theta^{a}}}+\frac{1}{I_{\theta^{b}}}} \\
i^{b} &= \frac{\frac{1}{I_{\theta^{b}}}}{\frac{1}{I_{\theta^{a}}}+\frac{1}{I_{\theta^{b}}}}
\end{aligned} \tag{8.8}$$

$$\begin{aligned}
\mathbf{q}^{a}\left(t\right) &= \mathrm{Slerp}\left(\hat{\mathbf{q}},\mathbf{q}_{correction},i^{a}\right)\mathbf{q}^{a}\left(t\right) \\
\mathbf{q}^{b}\left(t\right) &= \mathrm{Slerp}\left(\hat{\mathbf{q}},\overline{\mathbf{q}}_{correction},i^{b}\right)\mathbf{q}^{b}\left(t\right)
\end{aligned} \tag{8.9}$$

This involves quite a bit of work. To save time at the cost of accuracy, the magnitude of the inverse of the inertias can be used to compute i^a and i^b and applied to Equation (8.9) instead.

$$\begin{aligned}
i^{a} &= \frac{\left\|\mathbf{I}^{a^{-1}}\right\|}{\left\|\mathbf{I}^{a^{-1}}\right\|+\left\|\mathbf{I}^{b^{-1}}\right\|} \\
i^{b} &= \frac{\left\|\mathbf{I}^{b^{-1}}\right\|}{\left\|\mathbf{I}^{a^{-1}}\right\|+\left\|\mathbf{I}^{b^{-1}}\right\|}
\end{aligned} \tag{8.10}$$

When an angle limit is exceeded around any particular axis, the angular rates of the two rigid bodies around that axis must also be constrained.

$$\begin{aligned}\boldsymbol{\omega}^a_{world} &= \mathbf{q}^a(t)\boldsymbol{\omega}^a(t)\bar{\mathbf{q}}^a(t)\\ \boldsymbol{\omega}^b_{world} &= \mathbf{q}^b(t)\boldsymbol{\omega}^b(t)\bar{\mathbf{q}}^b(t)\\ \boldsymbol{\omega}_{correction} &= -(1+\varepsilon)\left(\boldsymbol{\omega}^a_{world}-\boldsymbol{\omega}^b_{world}\right)\end{aligned} \tag{8.11}$$

A coefficient of restitution, ε, is used to determine how fast the two bodies will rotate in the opposite direction after the limit is hit or whether they will simply stop rotating. The angular rate correction about any particular axis is set to zero if the orientation was not constrained around that axis. What is left of the angular rate correction is divided between the two rigid bodies using the same inertia ratios computed in Equations (8.8) or (8.10).

$$\begin{aligned}\boldsymbol{\omega}^a(t) &= \boldsymbol{\omega}^a(t)+i^a\left(\bar{\mathbf{q}}^a(t)\boldsymbol{\omega}_{correction}\mathbf{q}^a(t)\right)\\ \boldsymbol{\omega}^b(t) &= \boldsymbol{\omega}^b(t)-i^b\left(\bar{\mathbf{q}}^b(t)\boldsymbol{\omega}_{correction}\mathbf{q}^b(t)\right)\end{aligned} \tag{8.12}$$

9 Collision Constraint

Since the constraint iteration loop runs after the simulation loop (Section 10) it is possible for the effects of the constraints to overpower motion induced by some of the force models. This was observed in some of the early rag-doll simulations when parts of the rag-doll would penetrate into solid surfaces regardless of the collision model spring stiffness settings. The collision constraint solved this problem as the rag-doll's collision constraints are iterated along with the rest of its pin constraints. The rag-doll application was the only one that necessitated the use of collision constraints instead of collision impulses and forces.

As before, the collision detection algorithm returns a set of contact points, $\mathbf{c}_i$, surface normals, $\hat{\mathbf{n}}_i$, and the two rigid bodies involved in the collision. For each point and normal, the following algorithm is executed (all of the equations presented below should already be familiar).

$$\boldsymbol{\delta}\mathbf{x} = \left(\left(\mathbf{c}_i-\mathbf{e}^a_i\right)\cdot\hat{\mathbf{n}}^a_i\right)\hat{\mathbf{n}}^a_i \tag{9.1}$$

$$\begin{aligned}\mathbf{r}^a_i &= \mathbf{c}_i-\mathbf{x}^a(t)\\ \boldsymbol{\omega}^a_{world} &= \mathbf{q}^a(t)\boldsymbol{\omega}^a(t)\bar{\mathbf{q}}^a(t)\\ \mathbf{v}^a_{contact} &= \mathbf{v}^a(t)+\boldsymbol{\omega}^a_{world}\times\mathbf{r}^a_i\end{aligned} \tag{9.2}$$

$$\begin{aligned}\mathbf{r}^b_i &= \mathbf{c}_i-\mathbf{x}^b(t)\\ \boldsymbol{\omega}^b_{world} &= \mathbf{q}^b(t)\boldsymbol{\omega}^b(t)\bar{\mathbf{q}}^b(t)\\ \mathbf{v}^b_{contact} &= \mathbf{v}^b(t)+\boldsymbol{\omega}^b_{world}\times\mathbf{r}^b_i\end{aligned} \tag{9.3}$$

$$\mathbf{v}_{relative} = \mathbf{v}^a_{contact}-\mathbf{v}^b_{contact} \tag{9.4}$$

At this point, the desired velocity after the collision is computed.

$$
\begin{aligned}
\mathbf{v}_{perpendicular} &= \left(\mathbf{v}_{relative} \cdot \hat{\mathbf{n}}_i\right)\hat{\mathbf{n}}_i - \mathbf{v}_{relative} \\
\mathbf{v}^a_{new\ parallel} &= -\varepsilon\ \mathbf{v}^a_{relative} \cdot \hat{\mathbf{n}}_i \\
\mathbf{v}^a_{new\ perpendicular} &= \nu_{friction}\mathbf{v}_{perpendicular} \\
\mathbf{v}^a_{new} &= \mathbf{v}^a_{new\ parallel} + \mathbf{v}^a_{new\ perpendicular}
\end{aligned}
\tag{9.5}
$$

$$\boldsymbol{\delta}\mathbf{v} = \mathbf{v}^a_{new} - \mathbf{v}_{relative} \tag{9.6}$$

$\boldsymbol{\delta}\mathbf{x}$ from Equation (9.1) is partitioned between the two rigid bodies using the impulse equation.

$$
\begin{aligned}
\mathbf{r}^a_{body} &= \overline{\mathbf{q}}^a(t)\left(\mathbf{c}_i - \mathbf{x}^a(t)\right)\mathbf{q}^a(t) \\
\boldsymbol{\delta}\hat{\mathbf{x}}^a_{body} &= \overline{\mathbf{q}}^a(t)\,\boldsymbol{\delta}\hat{\mathbf{x}}\,\mathbf{q}^a(t) \\
\mathbf{r}^b_{body} &= \overline{\mathbf{q}}^b(t)\left(\mathbf{c}_i - \mathbf{x}^b(t)\right)\mathbf{q}^b(t) \\
\boldsymbol{\delta}\hat{\mathbf{x}}^b_{body} &= \overline{\mathbf{q}}^b(t)\,\boldsymbol{\delta}\hat{\mathbf{x}}\,\mathbf{q}^b(t)
\end{aligned}
\tag{9.7}
$$

$$j = \frac{-1}{\frac{1}{m^a} + \frac{1}{m^b} + \boldsymbol{\delta}\hat{\mathbf{x}}^a_{body} \cdot \left(\mathbf{I}^{a^{-1}}\left(\mathbf{r}^a_{body} \times \boldsymbol{\delta}\hat{\mathbf{x}}^a_{body}\right)\right) \times \mathbf{r}^a_{body} + \boldsymbol{\delta}\hat{\mathbf{x}}^b_{body} \cdot \left(\mathbf{I}^{b^{-1}}\left(\mathbf{r}^b_{body} \times \boldsymbol{\delta}\hat{\mathbf{x}}^b_{body}\right)\right) \times \mathbf{r}^b_{body}} \tag{9.8}$$

$$
\begin{aligned}
\mathbf{x}^a_{new}(t) &= \mathbf{x}^a(t) + \frac{j}{m^a}\boldsymbol{\delta}\mathbf{x} \\
\mathbf{j}^a_{body} &= \overline{\mathbf{q}}^a(t)\left(j\boldsymbol{\delta}\mathbf{x}\right)\mathbf{q}^a(t) \\
\boldsymbol{\delta}\mathbf{q}^a &= \text{ComputeQuaternion}\left(\mathbf{q}^a(t)\left(\mathbf{I}^{a^{-1}}\left(\mathbf{r}^a_{body} \times \mathbf{j}^a_{body}\right)\right)\overline{\mathbf{q}}^a(t)\right) \\
\mathbf{x}^b_{new}(t) &= \mathbf{x}^b(t) - \frac{j}{m^b}\boldsymbol{\delta}\mathbf{x} \\
\mathbf{j}^b_{body} &= \overline{\mathbf{q}}^b(t)\left(j\boldsymbol{\delta}\mathbf{x}\right)\mathbf{q}^b(t) \\
\boldsymbol{\delta}\mathbf{q}^b &= \text{ComputeQuaternion}\left(\mathbf{q}^b(t)\left(\mathbf{I}^{b^{-1}}\left(\mathbf{r}^b_{body} \times \mathbf{j}^b_{body}\right)\right)\overline{\mathbf{q}}^b(t)\right)
\end{aligned}
\tag{9.9}
$$

Similarly, $\boldsymbol{\delta}\mathbf{v}$ from Equation (9.6) is distributed between the two rigid bodies.

$$
\begin{aligned}
\boldsymbol{\delta}\hat{\mathbf{v}}^a_{body} &= \overline{\mathbf{q}}^a(t)\,\boldsymbol{\delta}\hat{\mathbf{v}}\,\mathbf{q}^a(t) \\
\boldsymbol{\delta}\hat{\mathbf{v}}^b_{body} &= \overline{\mathbf{q}}^b(t)\,\boldsymbol{\delta}\hat{\mathbf{v}}\,\mathbf{q}^b(t)
\end{aligned}
\tag{9.10}
$$

$$j = \frac{-1}{\frac{1}{m^a} + \frac{1}{m^b} + \boldsymbol{\delta}\hat{\mathbf{v}}^a_{body} \cdot \left(\mathbf{I}^{a^{-1}}\left(\mathbf{r}^a_{body} \times \boldsymbol{\delta}\hat{\mathbf{v}}^a_{body}\right)\right) \times \mathbf{r}^a_{body} + \boldsymbol{\delta}\hat{\mathbf{v}}^b_{body} \cdot \left(\mathbf{I}^{b^{-1}}\left(\mathbf{r}^b_{body} \times \boldsymbol{\delta}\hat{\mathbf{v}}^b_{body}\right)\right) \times \mathbf{r}^b_{body}} \tag{9.11}$$

$$\begin{aligned}
\mathbf{v}^a_{new}(t) &= \mathbf{v}^a(t) + \frac{j}{m^a}\boldsymbol{\delta}\mathbf{v} \\
\boldsymbol{\omega}^a(t) &= \boldsymbol{\omega}^a(t) + \mathbf{I}^{a^{-1}}\left(\mathbf{r}^a_{body} \times \left(\overline{\mathbf{q}}^a(t)\left(j\boldsymbol{\delta}\mathbf{v}\right)\mathbf{q}^a(t)\right)\right) \\
\mathbf{v}^b_{new}(t) &= \mathbf{v}^b(t) - \frac{j}{m^b}\boldsymbol{\delta}\mathbf{v} \\
\boldsymbol{\omega}^b(t) &= \boldsymbol{\omega}^b(t) - \mathbf{I}^{b^{-1}}\left(\mathbf{r}^b_{body} \times \left(\overline{\mathbf{q}}^b(t)\left(j\boldsymbol{\delta}\mathbf{v}\right)\mathbf{q}^b(t)\right)\right)
\end{aligned} \tag{9.12}$$

As before, the quaternions are updated last.

$$\begin{aligned}
\mathbf{q}^a(t) &= \boldsymbol{\delta}\mathbf{q}^a\mathbf{q}^a(t) \\
\mathbf{q}^b(t) &= \boldsymbol{\delta}\mathbf{q}^b\mathbf{q}^b(t)
\end{aligned} \tag{9.13}$$

10 Simulation Loop

The simulation loop is the main function that executes all of the force models, equations of motion and constraint solvers. The simulation runs at a minimum frequency for stability, so the entire simulation loop must be executed multiple times for larger step sizes. This minimum frequency depends on the force models used and is not a function of the constraints. In games, $f_{simulation}$ is typically 30 or 60 Hz. The number of iterations is computed using Equation (10.1).

$$n_{simulation} = \lfloor f_{simulation}\Delta t \rfloor + 1 \tag{10.1}$$

As the number of constraint iterations is increased, the net residual error decreases and the constraints appear stronger. The number of iterations does not have to remain constant. It can be adjusted dynamically as a function of distance to the camera, for example. Conversely, the number of iterations can be decreased to make objects appear more flexible. For most applications, two or three iterations are sufficient.

The ordering of the constraints is also critical. Every constraint that gets solved is affected by the constraints that follow, while the last constraint remains unaffected. To strengthen constraints, move them down in the list, to weaken constraints, move them up. Likewise, motion introduced by a force model can be overridden by a constraint. The simulation loop is illustrated in Figure 5.

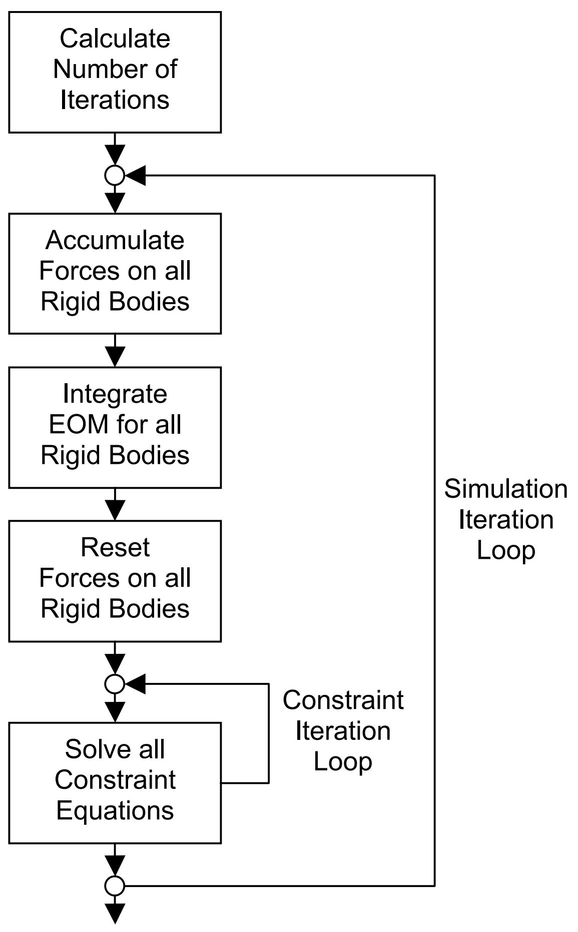
Calculate Number of Iterations
Accumulate Forces on all Rigid Bodies
Integrate EOM for all Rigid Bodies
Reset Forces on all Rigid Bodies
Solve all Constraint Equations
Constraint Iteration Loop
Simulation Iteration Loop

Figure 5 - Simulation Loop

11 Rag-doll Simulation

The rag-doll simulation used in the game MX Unleashed, published by THQ, uses 18 rigid bodies, 17 pin, 17 angle and 9 collision constraints. The number of body parts can vary, but is usually determined by the needs of the animator. Spheres centered on some, but not all, of the body parts, were used to detect collisions. Depending on the performance needs of the application, spheres or the body parts themselves can be used. The setup used in MX Unleashed is illustrated in Figure 6.

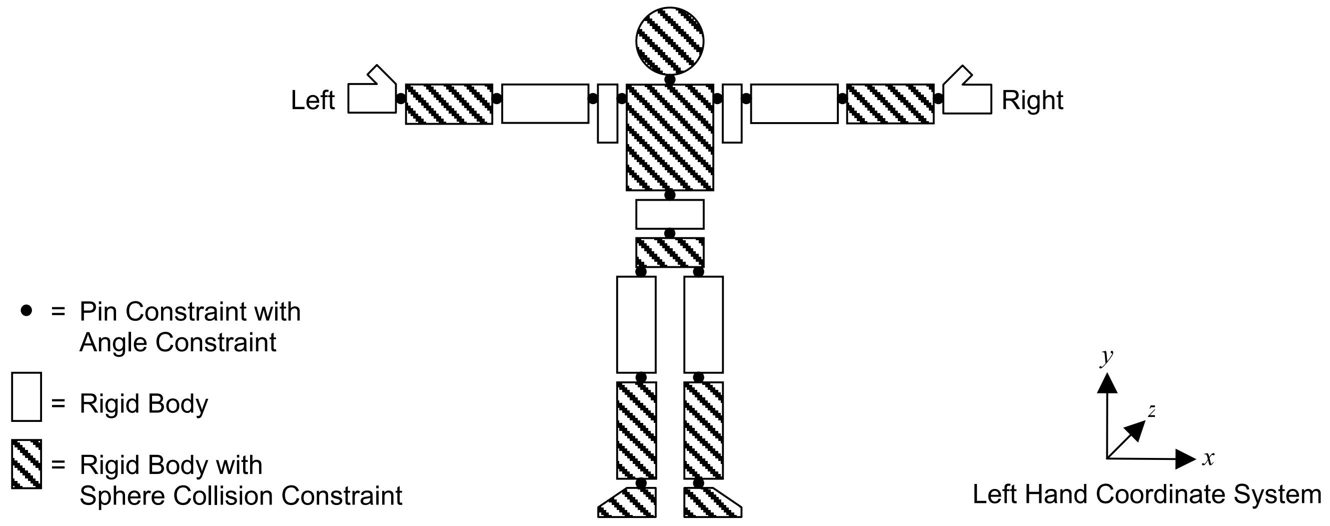
Left
Right
● = Pin Constraint with Angle Constraint
▭ = Rigid Body
▨ = Rigid Body with Sphere Collision Constraint
y
z
x
Left Hand Coordinate System

Figure 6 - Rag-doll Simulation Used in MX Unleashed

The positions of the pin constraints relative to the rigid bodies they connected were extracted from the geometry generated by the animator. The center of gravity was placed in the geometric center of each body part. The mass, inertias and angle limits are presented in Table 1. These mass properties are clearly not indicative of a real human body. Instead, they are settings that generated the desired motion in our game. The rag-doll simulation runs at 30 Hz (one pass through the simulation) using two iterations of the constraint equations.

Body Part	m (slug)	$\mathbf{I}$ (slug ft^2)	Θ_{min} (degrees)	Θ_{max} (degrees)
Pelvis	0.35	(1, 1, 1)	N/A	N/A
Lower Torso	0.25	(1, 1, 1)	(0, -20, -10)	(20, 20, 10)
Upper Torso	0.20	(1, 1, 1)	(0, -20, -10)	(20, 20, 10)
Head	0.20	(1, 1, 1)	(-35, -50, -40)	(45, 50, 40)
Left Shoulder	0.25	(1, 1, 1)	(-45, 0, 0)	(0, 0, 0)
Left Upper Arm	0.25	(1, 1, 1)	(0, 0, -10)	(90, 120, 30)
Left Lower Arm	0.20	(1, 1, 1)	(-45, 0, 0)	(0, 110, 0)
Left Hand	0.20	(1, 1, 1)	(0, -20, -45)	(90, 20, 45)
Right Shoulder	0.25	(1, 1, 1)	(-45, 0, 0)	(0, 0, 0)
Right Upper Arm	0.25	(1, 1, 1)	(0, -120, -30)	(90, 0, 10)
Right Lower Arm	0.20	(1, 1, 1)	(-45, -110, 0)	(0, 0, 0)
Right Hand	0.20	(1, 1, 1)	(-90, -20, -45)	(0, 20, 45)
Left Upper Leg	0.25	(1, 1, 1)	(-60, 0, 0)	(0, 0, 45)
Left Lower Leg	0.20	(1, 1, 1)	(0, 0, 0)	(135, 0, 0)
Left Foot	0.15	(1, 1, 1)	(-20, -20, 0)	(45, 20, 0)
Right Upper Leg	0.25	(1, 1, 1)	(-60, 0, 0)	(0, 0, 45)
Right Lower Leg	0.20	(1, 1, 1)	(0, 0, 0)	(135, 0, 0)
Right Foot	0.15	(1, 1, 1)	(-20, -20, 0)	(45, 20, 0)

Table 1 - Rag-doll Mass Properties and Angle Constraints

12 Performance

The rag-doll simulation described in Section 11 was run on both the PlayStation 2 and XBox. On the PS2, the equations of motion and the constraint equations run entirely on the vector unit (VU0) in macromode. The simulation loop runs on the CPU. On the XBox, the quaternion math was hand optimized using the SSE (Streaming SIMD Extensions) instruction set but the compiler optimized the rest of the code. The timing results are listed in Table 2. The collision detection times are not included in these results. However, the collision constraint times include a small percentage of time that should actually be attributed to the collision detection algorithm.

Algorithm	Number of Calls	Total Time **PS2** (mSec)	Time Per Call **PS2** (mSec)	Total Time **XBox** (mSec)	Time Per Call **XBox** (mSec)
Pin Constraint	34 (2 x 17)	0.17	0.0051	0.15	0.0044
Angle Constraint	34 (2 x 17)	0.15	0.0045	0.21	0.0062
Collision Constraint	18 (2 x 9)	0.15	0.0086	0.13	0.0072
Equations of Motion	18 (1 x 18)	0.03	0.0017	0.06	0.0033
Total		**0.50**		**0.55**	

Table 2 - Rag-doll Performance on the PlayStation 2 and XBox

13 Conclusion

The Verlet-like integrator made it possible to integrate systems of sufficient stiffness for our applications. This removed the need to use implicit integrators which can be difficult to program and expensive to solve. Penalty methods could then be used to resolve resting forces instead of having to solve for them explicitly. The difficulty in getting rigid bodies to sit still on a surface when using the impulse method was also avoided.

The addition of the rotational degrees of freedom simplified the simulation of rigid bodies and made more traditional pin and angle constraints possible. Replacing the last position and quaternion with a velocity and angular rate solved the problem of computing damping forces based on a noisy difference equation. However, this made the constraint equations more expensive as a result.

By trading the stability associated with implicit integration for performance, we feel that this simulation is the ideal trade-off for this generation of console games. As memory and CPU speeds increase, it may not be as critical to keep the code as fast as possible and implicit integration using large sparse matrices may become the best solution for the future.

Please feel free to contact the authors should you have any questions, corrections, suggestions or improvements.

References

[Witkin]	Witkin, Andrew. Constrained Dynamics. SIGGRAPH 1999 Course Notes, pages F1-F12.
[Jakobsen]	Jakobsen, Thomas. Advanced Character Physics. Game Developers Conference 2001 Proceedings, pages 383-401. CMP Media Inc.
[Garberoglio]	Garberoglio, Giovanni. Dynamical Properties Of H-Bonded Liquids: A Theoretical and Computer Simulation Study. Faculty of Physical and Natural Mathematical Sciences. University of Trento. 2001. http://garmi.science.unitn.it/~gio/PhD_thesis/
[Refson]	Refson, Keith. Moldy Users Manual. Department of Earth Sciences. Oxford University. 2001. http://www.earth.ox.ac.uk/~keithr/moldy-manual/moldy.html
[Baraff]	Baraff, David. Rigid Body Simulation. SIGGRAPH 1999 Course Notes, pages D1-D68.

Appendix A – Quaternion Operations

Basic quaternion:

$$\mathbf{q} = \begin{bmatrix} q_w \\ q_x \\ q_y \\ q_z \end{bmatrix}$$

Unit quaternion:

$$\hat{\mathbf{q}} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

Conjugate:

$$\overline{\mathbf{q}} = \begin{bmatrix} q_w \\ -q_x \\ -q_y \\ -q_z \end{bmatrix}$$

Inner (Dot) Product:

$$\mathbf{q}^a \cdot \mathbf{q}^b = q_w^a q_w^b + q_x^a q_x^b + q_y^a q_y^b + q_z^a q_z^b$$

Norm:

$$|\mathbf{q}| = \sqrt{q_w q_w + q_x q_x + q_y q_y + q_z q_z}$$

Multiplication:

$$\mathbf{q}^a \mathbf{q}^b = \begin{bmatrix} q_w^a & -q_x^a & -q_y^a & -q_z^a \\ q_x^a & q_w^a & -q_z^a & q_y^a \\ q_y^a & q_z^a & q_w^a & -q_x^a \\ q_z^a & -q_y^a & q_x^a & q_w^a \end{bmatrix} \begin{bmatrix} q_w^b \\ q_x^b \\ q_y^b \\ q_z^b \end{bmatrix}$$

First Derivative:

$$\dot{\mathbf{q}}(\mathbf{q},\boldsymbol{\omega}) = \frac{1}{2}\mathbf{q}\begin{bmatrix} 0 \\ \omega_x \\ \omega_y \\ \omega_z \end{bmatrix}$$

Second Derivative:

$$\ddot{\mathbf{q}}(\mathbf{q},\boldsymbol{\omega},\dot{\boldsymbol{\omega}}) = \frac{1}{2}\mathbf{q}\begin{bmatrix} -2\dot{\mathbf{q}}\cdot\dot{\mathbf{q}} \\ \dot{\omega}_x \\ \dot{\omega}_y \\ \dot{\omega}_z \end{bmatrix}$$

Conversion to Angle Axis Vector:

$$\text{ComputeAngleAxis}(\mathbf{q}) = \frac{\theta}{\sin\frac{\theta}{2}}\begin{bmatrix} q_x \\ q_y \\ q_z \end{bmatrix}$$

where

$$\theta = 2\cos^{-1} q_w$$

Conversion from Angle Axis Vector:

$$\text{ComputeQuaternion}(\mathbf{v}) = \frac{1}{\theta}\begin{bmatrix} \theta\cos\frac{\theta}{2} \\ v_x \sin\frac{\theta}{2} \\ v_y \sin\frac{\theta}{2} \\ v_z \sin\frac{\theta}{2} \end{bmatrix}$$

where

$$\theta = |\mathbf{v}|$$

Spherical Linear Interpolation:

$$\text{Slerp}\left(\mathbf{q}^a, \mathbf{q}^b, t\right) = \begin{bmatrix} \alpha q_w^a + \beta q_w^b \\ \alpha q_x^a + \beta q_x^b \\ \alpha q_y^a + \beta q_y^b \\ \alpha q_z^a + \beta q_z^b \end{bmatrix}$$

where

$$\alpha = \frac{\sin\left((1-t)\theta\right)}{\sin\theta}$$

$$\beta = \frac{\sin\left(t\theta\right)}{\sin\theta}$$

$$\theta = \cos^{-1}\left(\mathbf{q}^a \cdot \mathbf{q}^b\right)$$